VME Clock Timer Board

Analog control of AMD9513 channels and clock events

R. Goodwin

Oct 10, 1989

The VME clock timer board uses four AMD9513 chips each of which provides two 32-bit timers with one μ sec resolution in the settable delays. It also includes a 256-byte memory which specifies which of the channels (up to eight) is triggered for each of the 256 possible clock events. Each bit position in the byte corresponds to a different timing channel. This note describes how the VME software implements control and readout of these timing channels.

Design

The generic parameter page is typically used for display and control of numeric values via analog channels. When new devices are added to the system, it is worth trying to see if they can fit into this generic scheme and thus inherit the features that parameter page interaction provides. It also serves to present a consistent interface to the user. The features of the clock timer board that require support for numeric values are these:

- Read and write the 32-bit delay settings (timing channels)
- Set possible multiple clock events which can trigger the delays
- Read what clock events are currently set for each timing channel

Since the delay values are 32-bit quantities, and the VME software does not support 32-bit setting values, the timer delay channels are represented by a pair of analog channels which include a “coarse” and a “fine” control. The coarse channel is in units of msec, and the fine channel is in units of μ sec. On a parameter page, it is easy to read out this way, especially if the engineering units are chosen as msec for both coarse and fine channels. For timing to only msec precision, adjustment of the coarse channel is sufficient. Fine tuning is possible down to the hardware resolution by use of the fine control channel. This coarse and fine treatment also helps to solve the knob resolution problem inherent with adjustment of hi resolution devices.

The use of coarse and fine channels as described above allows for more than one pair of coarse and fine values that can result in the same time delay. Since the counters can be read back to provide a reading of the delay value, and since that conversion into coarse and fine values in *not* ambiguous, the setting ambiguity is resolved by allowing the fine channel to carry into or borrow from the coarse channel when the end of its range is reached.

With the fine channel using μ sec and the coarse channel using msec, the limit in the range of delay setting is only about 32 seconds (if we keep the values positive). If this is insufficient for the users’ needs, then the coarse channel could be changed to have units of 10 msec, which would allow for a range of delay setting of about 5 minutes.

setting values is a clock event number. The number of such channels to be installed can be tailored to the needs of the users based upon the intended use of the timing channel and on how the entire clock system is organized. When a clock event setting is made to one of these channels, it checks to see whether that event is already selected for that timing channel. If it is, the setting is ignored. If it is not, the current event setting of the channel to be set is first deselected, and then the new event is selected. This insures that all such setting channels will have unique event number setting values. Note that this logic means that one can adjust an event number with the knob and end up with only the last clock event selected. A zero value setting can be used to deselect an event that was set before. But since there is actually a clock event #0, the value 256 can be used to cause event#0 to be set.

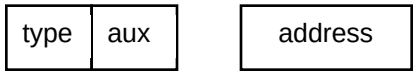
As a shorthand means of clearing selected clock events, a special setting is allowed that causes clearing of all selected events from the 256-byte RAM for a given timing channel. At system reset time, when the restore of D/A settings is taking place in channel number order, the channels holding clock event values will reselect those events for a given timing channel. This means that a *lower-numbered* channel should be used for the purpose of clearing all selected events for that timing channel.

To provide a readout of the selected events that were set, one could merely copy the setting value to the reading. This would partly work, but it has difficulty with the “clear all” setting just described. To get around this, a reading of these event channels supplies a copy of the last setting only as long as the corresponding RAM bit remains set. As soon as it is cleared, by whatever reason, the event channel will have its setting cleared, and the reading will then become zero also.

The readout for a “clear all” channel is a count of the number of clock events selected for a given timing channel. This channel could be scanned for alarms to insure that the number of selected events did not change unexpectedly. To execute the “clear all” function, a zero value must be used for the setting. Then the reading would naturally be expected to become zero as well.

Analog control field

The specification of the control parameters needed for 9513 channel control must be contained in the Analog Control field of the analog descriptor. The format of that 4-byte field is as follows:



The first byte indicates the type of analog control, which in this case would use the value `$0F` to specify 9513 timer type. The meaning of the other three bytes is dependent upon the type, so we are free to choose anything convenient. Let the aux byte specify which timer channel, whether it is the coarse or fine word, and whether it is a delay setting, an event setting or a clear-all-events setting. The address word is enough to indicate the board's base address as it is presumed to be in VME short-I/O space, which means that the upper part of the address is $FFFF. (In fact, the base address must be on a 4K boundary, which means that only 4 bits is really needed to specify the board's VME base address.)

Specific values of the aux byte are:

`$0x` Coarse delay chan #`x`
`$1x` Fine delay chan #`x`
`$2x` Set event# for chan #`x`
`$3x` Clear all events for chan #`x`

The value of the x nibble is 0–1 for the prototype board. It will range from 0–7 for the final board, since that board supports 8 timing channels. Each timing channel will require four analog channels (or more, in order to handle multiple selected events) to cover all choices.

The following steps are performed to set the two timer channels on the prototype board:

Chan #0	Chan #1
$C100 → (17)	$C100 → (17)
$01E1 → (01)	$03E1 → (03)
$1221 → (02)	$1421 → (04)
$0000 → (09)	$0000 → (0B)
lsw → (11)	lsw → (13)
msw* → (0A)	msw* → (0C)
(63) (6C)	

The numbers in parentheses are register selects at the byte at offset $810 from the board's base address, followed by an access to the ls byte and then the ms byte to make up the word, each referencing the byte at offset $800 from the base address.

When a setting is made to a delay channel, the entire initialization logic will be performed the first time a delay is written. The coarse and fine channels should be arranged to be consecutive channels so the setting software can put the two halves together.

The lsw and msw* refer to the lo order and modified hi order words of the 32-bit setting. The last step does the Load and Arm operation. If the Master Mode register is read and found to not be $C100, the entire chip must be initialized. If the a timing channel's mode register is read and has the value not ending in $E1, then that counter must be initialized as above. If the Master Mode register looks good, and the counter's mode register also looks good, the only the lsw and then the msw* words should be written to set the counter delay. The other steps should be skipped.

Internals

A new analog setting routine called `SET9513` was written to support all settings for this board. It is invoked when the analog control type byte=`$0F`. It uses the aux byte and the addr word to do its I/O, including initializing the 9513 chip when necessary.

Three new data access table routines were written to support readings of the types mentioned here. Type `$17` invokes `RDEVCNT` to tally the number of clock events selected for a given set of timing channels on one board. Type `$18` invokes `RDEVNTS` to test whether the last selected event is still selected in the trigger RAM. Type `$19` invokes `RD9513D` to read back the coarse and fine delay counts.

All four new routines are contained in the `MOD9513` module of 500 lines of code which assembles to about 900 bytes.